//1/

#include <iostream>

using namespace std;

struct node{

int num;

node\* left;

node\* right;

};

void addNode(node\* nod, int data){

if (data < nod->num)

if (nod->left)

addNode(nod->left, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->left = newNode;

}

else

if (nod->right)

addNode(nod->right, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->right = newNode;

}

}

void UnorderedBinaryTree (node\* nod, int u, int v, int \*UBTsum){

if (!nod)

return;

cout<< "nod->num = "<<nod->num<<endl<<endl;

if (nod->num>u && nod->num< v){

\*UBTsum = \*UBTsum+nod->num;

//cout << "UBTsum = "<<\*UBTsum<< endl<<endl;

cout<<"its number which u need = "<<nod->num<< endl;

}

if (nod->left)

UnorderedBinaryTree(nod->left, u, v, UBTsum);

if (nod->right )

UnorderedBinaryTree(nod->right, u, v, UBTsum);

}

void OrderedBinaryTree (node\* nod, int u, int v, int \*OBTsum){

if (!nod)

return;

if (nod->num>u)

OrderedBinaryTree(nod->left, u, v, OBTsum);

cout<< "nod->num = "<<nod->num<<endl;

if (nod->num>u && nod->num<v){

cout<<"its number which u need = "<<nod->num<< endl<<endl;

\*OBTsum = \*OBTsum+nod->num;

}

if (nod->num<v )

OrderedBinaryTree(nod->right, u, v, OBTsum);

}

int main(){

int arr[14] = {9,14,5,7,13,16,3,12,20,4,11,6,15,16};

node\* root = new node;

root->left = root->right = NULL;

root->num = 10;

for (int i = 0; i < 14; i++)

addNode(root, arr[i]);

// UnorderedBinaryTree

cout<<"UBT"<< endl;

cout<<"U=6"<<" "<<"V=15"<<endl;

int UBTsum = 0;

UnorderedBinaryTree (root, 6, 15, &UBTsum);

cout<< endl<<"UBTsum = "<< UBTsum;

cout<< endl<< endl<< endl;

// OrderedBinaryTree

cout<< endl<<"OBT"<<endl;

cout<<"U=6"<<" "<<"V=15"<<endl;

int OBTsum = 0;

OrderedBinaryTree (root, 6, 15, &OBTsum);

cout<< endl<<"OBTsum = "<< OBTsum;

return 0;

}

//2/  
empty

//3/

#include <iostream>

using namespace std;

struct node{

int num;

node\* left;

node\* right;

};

void addNode(node\* nod, int data){

if (data < nod->num)

if (nod->left)

addNode(nod->left, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->left = newNode;

}

else

if (nod->right)

addNode(nod->right, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->right = newNode;

}

}

//find the largest value

void FTLV (node\* nod, int v, int \*biggestNUM){

if (!nod)

return;

if (nod->num<v)

FTLV(nod->left, v, biggestNUM);

cout<< "nod->num = "<<nod->num<<endl;

if ( nod->num<v && \*biggestNUM<v){

cout<<"this number is largest than the previous = "<<nod->num<< endl<<endl;

\*biggestNUM = nod->num;

}

if (nod->num<v )

FTLV(nod->right, v, biggestNUM);

}

int main(){

int arr[14] = {9,14,5,7,13,16,3,12,20,4,11,6,15,16};

node\* root = new node;

root->left = root->right = NULL;

root->num = 10;

for (int i = 0; i < 14; i++)

addNode(root, arr[i]);

cout<< endl<<"find the largest value before 'v'"<<endl;

cout<<"V=17"<<endl;

int biggestNUM = 0;

FTLV (root, 17, &biggestNUM);

cout<< endl<<"the largest value = "<< biggestNUM;

return 0;

}

//4/

#include <iostream>

using namespace std;

struct node{

int num;

node\* left;

node\* right;

};

void addNode(node\* nod, int data){

if (data < nod->num)

if (nod->left)

addNode(nod->left, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->left = newNode;

}

else

if (nod->right)

addNode(nod->right, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->right = newNode;

}

}

//find the way

void find\_the\_way (node\* nod, int i, int v, int \*TrackArray, int j, bool& informer){

if (!nod)

return;

cout <<"num\_move / num = " << i<<" / "<<nod->num<<endl;

i++;

TrackArray[j]=nod->num;

j++;

if (nod->num==v)

{

informer = true;

cout<<endl<<endl<< "V exist"<<endl;

}

if (nod->left && nod->num > v)

find\_the\_way(nod->left,i, v, TrackArray, j, informer);

if (nod->right && nod->num <v )

find\_the\_way(nod->right,i, v, TrackArray, j, informer);

}

void showBinary(node\* nod){

if (nod)

{

showBinary(nod->left);

cout << nod->num << "; ";

showBinary(nod->right);

}

}

int main(){

int TrackArray[10]= {};

int arr[14] = {9,14,5,7,13,16,3,12,20,4,11,6,15,17};

node\* root = new node;

root->left = root->right = NULL;

root->num = 10;

for (int i = 0; i < 14; i++)

addNode(root, arr[i]);

bool informer = false;

int i = 0;

int j=0;

int v= 11;

cout<<"V = "<<v<<endl;

showBinary(root);

cout<<endl;

find\_the\_way (root,i, v ,TrackArray, j, informer);

if (informer)

{

int newSize = 0;

for (int i = 0; i < 10; i++)

{

if (TrackArray[i] != 0)

{

TrackArray[newSize] = TrackArray[i];

newSize++;

}

}

for (int i = 0; i < newSize; i++)

{

if (i==0) cout << "root = " << TrackArray[i] << endl;

if (i!=0 && TrackArray[i-1] > TrackArray[i]) cout << "go left to " << TrackArray[i] << endl;

if (i!=0 && TrackArray[i-1] < TrackArray[i]) cout << "go right to " << TrackArray[i] << endl;

}

}

else cout << "I didnt find V";

return 0;

}